

0002000-1001  
TOP SECRET

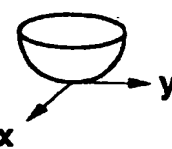
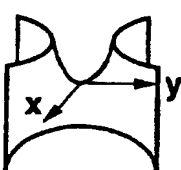
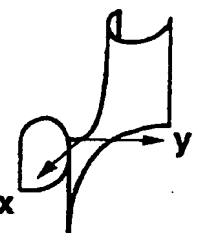




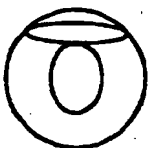


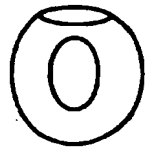
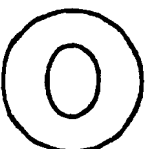
INDEX	0	1	1	2
EXPRESSION OF SINGULAR POINTS	$x^2 + y^2$	$-x^2 + y^2$	$x^2 - y^2$	$-x^2 - y^2$
SHAPE OF SINGULAR POINT NEIGHBORHOOD				
	GIVE 0 CELL	GIVE 1 CELL	GIVE 1 CELL	GIVE 2 CELLS
	↓	↓	↓	↓
EQUIVALENT CELL COMPLEX				
SET OF POINTS BELOW CROSS SECTION				

FIG. 1

FIG. 2(a) FIG. 2(b) FIG. 2(c)

FIG. 2(a)

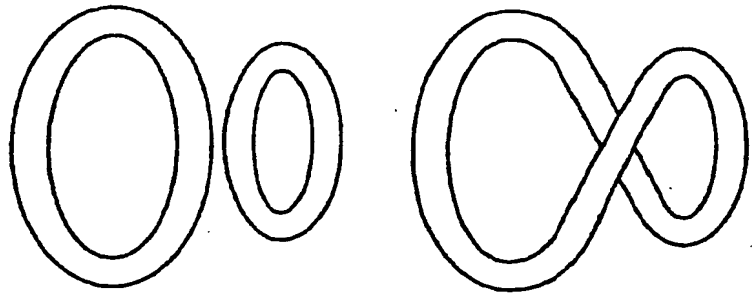


FIG. 2(b)

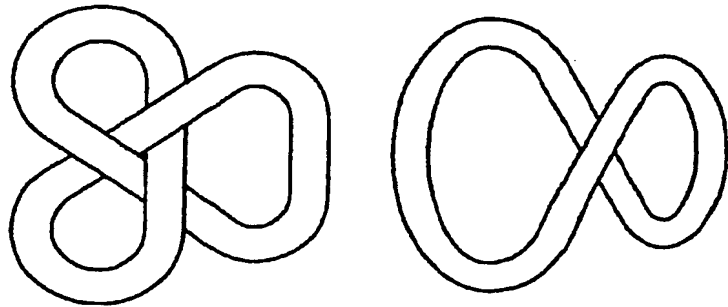
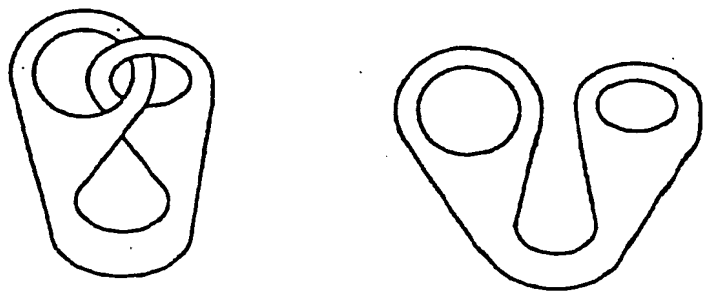


FIG. 2(c)



120

VERTEX

SADDLE  
POINT

SADDLE  
POINT

PIT

FIG. 3(a)

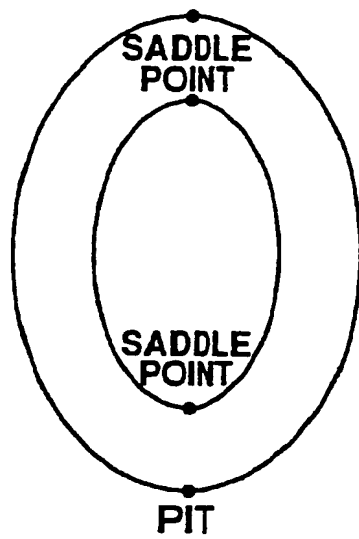


FIG. 3(b)

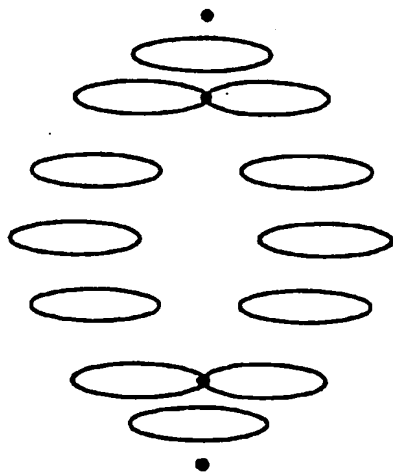
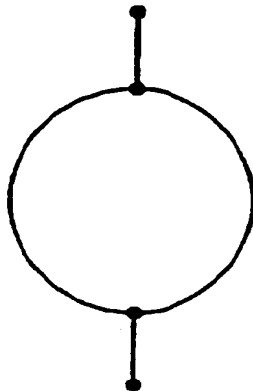


FIG. 3(c)



00072025-101001  
TOP SECRET 5262660

FIG. 4(a)

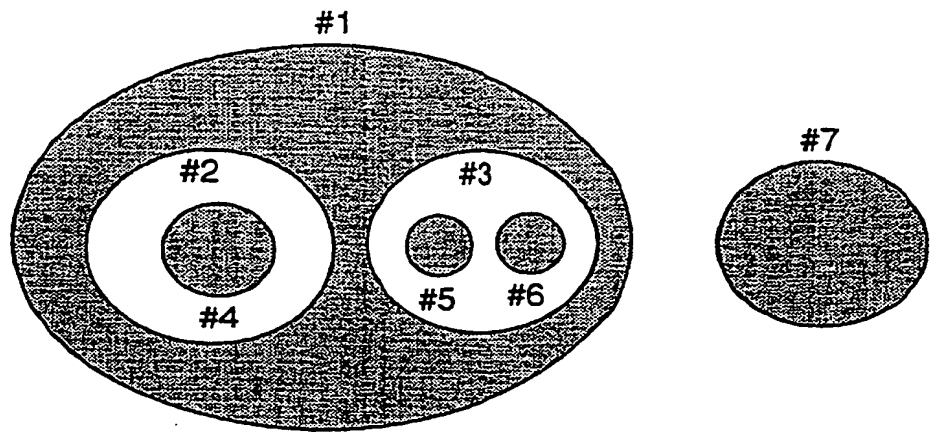
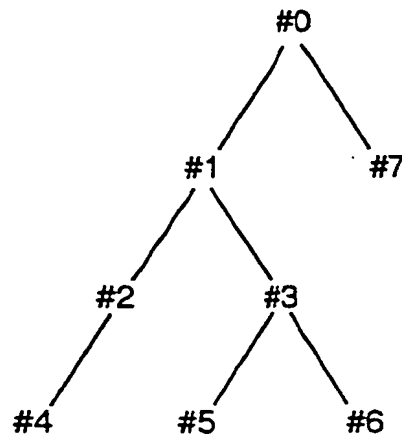


FIG. 4(b)



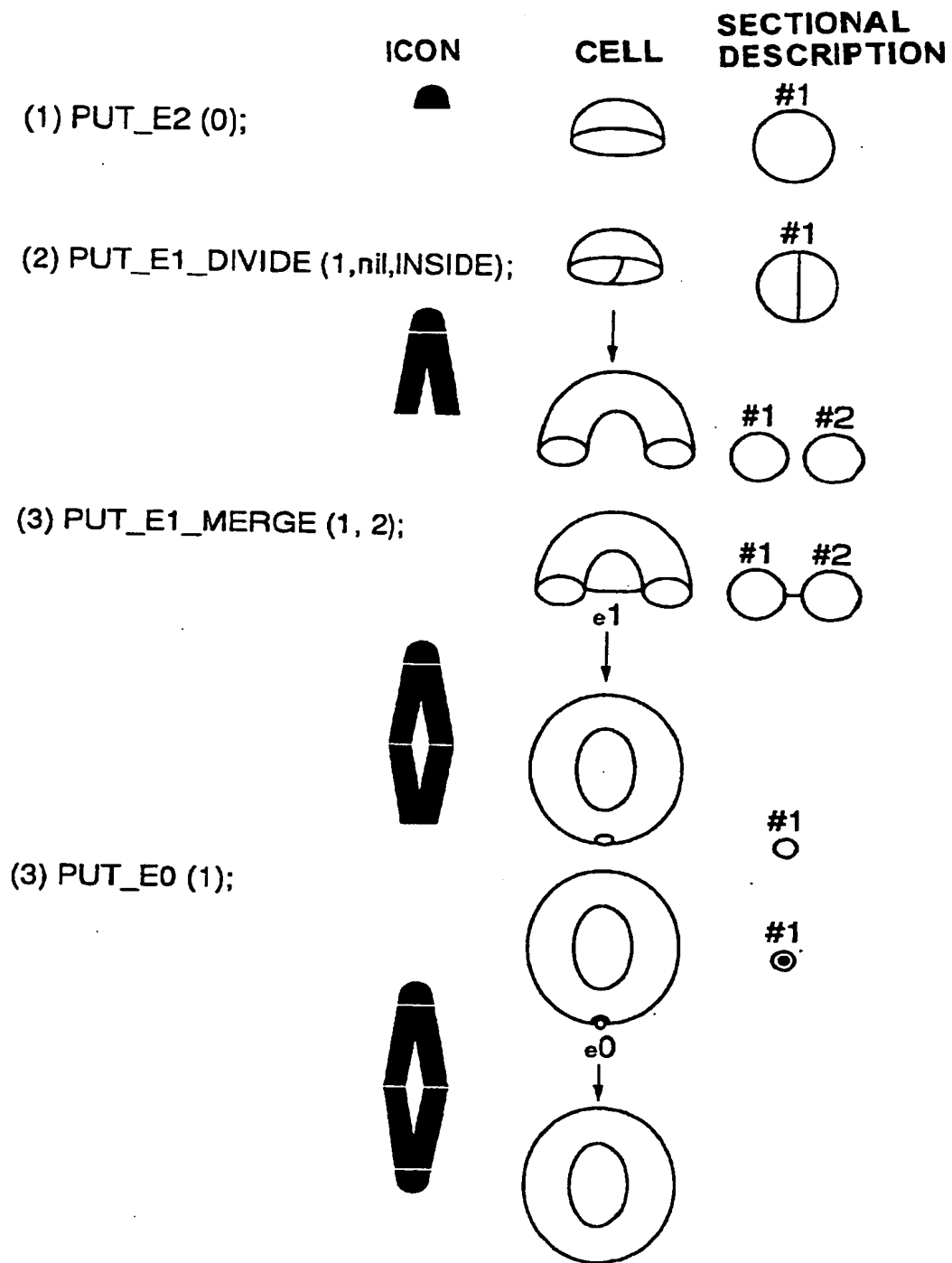


FIG. 5

```

program operators(input, output);
constant
    enabled = true;
    disabled = false;
    inside = true;
    outside = false;
    end_of_list = -1;

type
    contour_number = 0..max_contour_number;
    child_list = array[1..maxchildren] of contour_number;
    pointer_to_child_list = ↑ child_list;

var
    children: array[contour_number] of pointer_to_child_list;
    parent#: array[contour_number] of contour_number;
    number_of_children: array[contour_number] of integer;
    most_recently_created#: contour_number;
    contour_status: array[contour_number] of boolean;

```

FIG. 6

```

procedure add_listed_children(n:contour_number;clist:pointer_to_child_list);
    {details are omitted}
procedure remove_listed_children(n:contour_number;clist:pointer_to_child_list);
    {details are omitted}
function are_children(n:contour_number;clist:pointer_to_child_list);boolean;
    {details are omitted}
function in_list(n:contour_number;clist:pointer_to_child_list);boolean;
    {details are omitted}
function list_containing_only(n:contour_number):pointer_to_child_list;
var
    n_as_list: pointer_to_child_list;
begin
    new(n_as_list);
    n_as_list ↑ [1] := n;
    n_as_list ↑ [2] := end_of_list;
    list_containing_only := n_as_list;

end;

```

FIG. 7

```

a
procedure put_e2(n: contour_number);
begin
    if (contour_status[n] = disabled) then go to error;
    create_new_contour;
    add_listed_children(n, list_containing_only(most_recently_created#));
end;

b
procedure put_e0(n: contour_number);
begin
    if ((contour_status[n] = disabled) or not all_successor_disabled(n))
        then goto error;
    contour_status[n] := disabled;
end;

c
procedure put_e1_divide(n: contour_number; clist: pointer_to_child_list; inside: boolean);
begin
    if ((contour_status[n] = disabled) or (contour_status[parent#[n]] = disabled))
        then goto error;
    create_new_contour;
    add_listed_children(most_recently_created#, clist);
    if (not inside and are_children(parent#[n], clist)
        and not in_list(n, list)) or (clist = nil))
    then begin
        remove_listed_children(parent#[n], clist);
        add_listed_children(n, list_containing_only(most_recently_created#));
    end
    else if (inside and (are_children(n, clist) or (clist = nil)))
    then begin
        remove_listed_children(n, clist);
        add_listed_children(parent#[n], list_containing_only(most_recently_created#));
    end
    else go to error;
end;

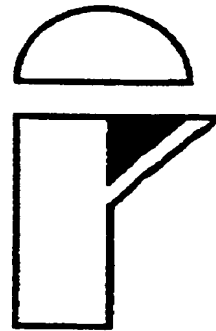
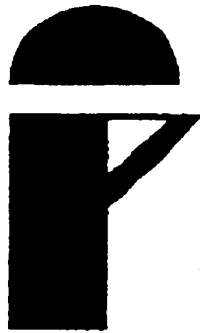
d
procedure put_e1_merge(c1: contour_number; c2: contour_number);
begin
    if ((contour_status[c1] = disabled) or (contour_status[c2] = disabled))
        then goto error;
    if (c1 = parent#[c2]) then
        add_listed_children(parent#[c1], children[c2]);
    else if (parent#[c1] = parent#[c2]) then
        add_listed_children(c1, children[c2]);
    else go to error;
    remove_listed_child(parent#[c2], list_containing_only(c2));
    contour_status[c2] := disabled;
end;

```

FIG. 8

# HOLLOW

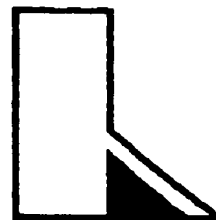
## 2 CELLS



## Put\_e1\_merge



## Put\_e1\_merge



## Put\_e1\_divide



## Put\_e1\_divide

**FIG. 9**

[illegible]



TOP SECRET

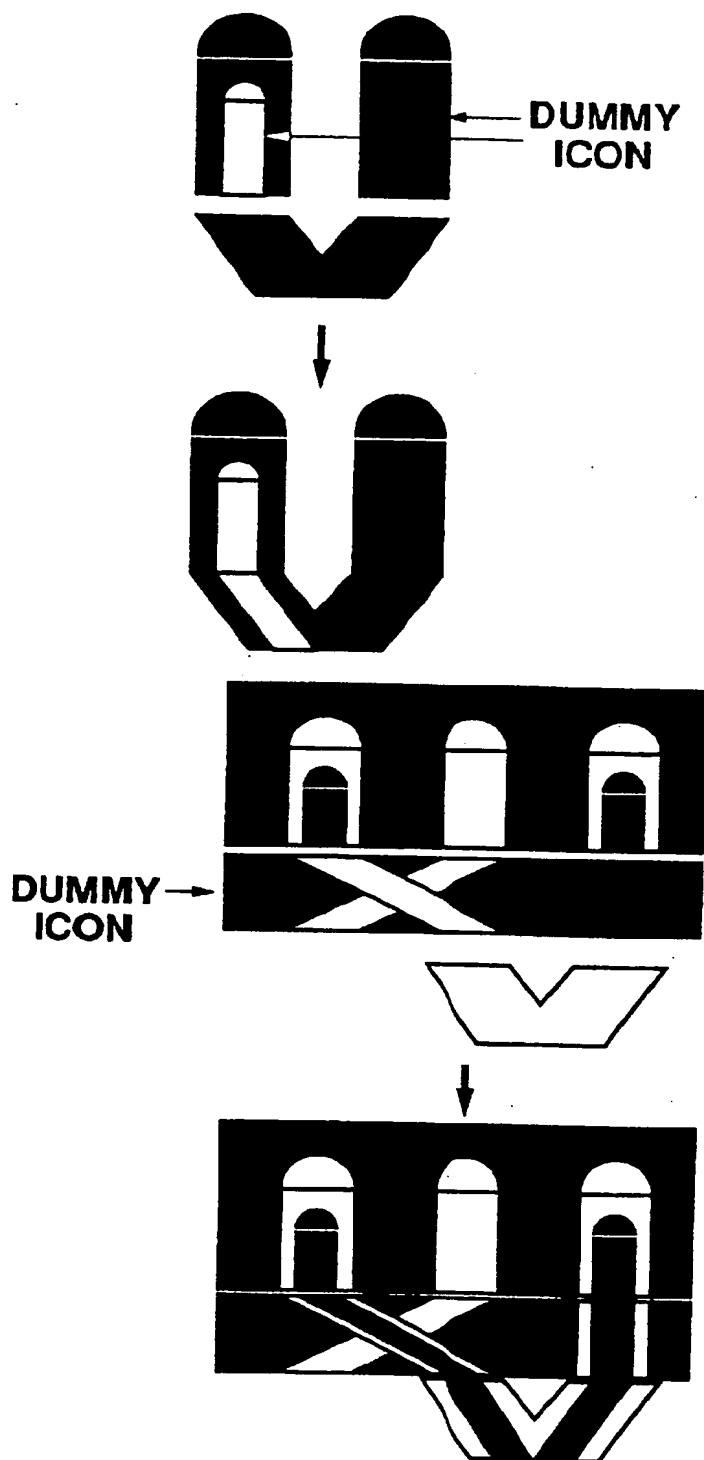


FIG. 10

FIG. 11

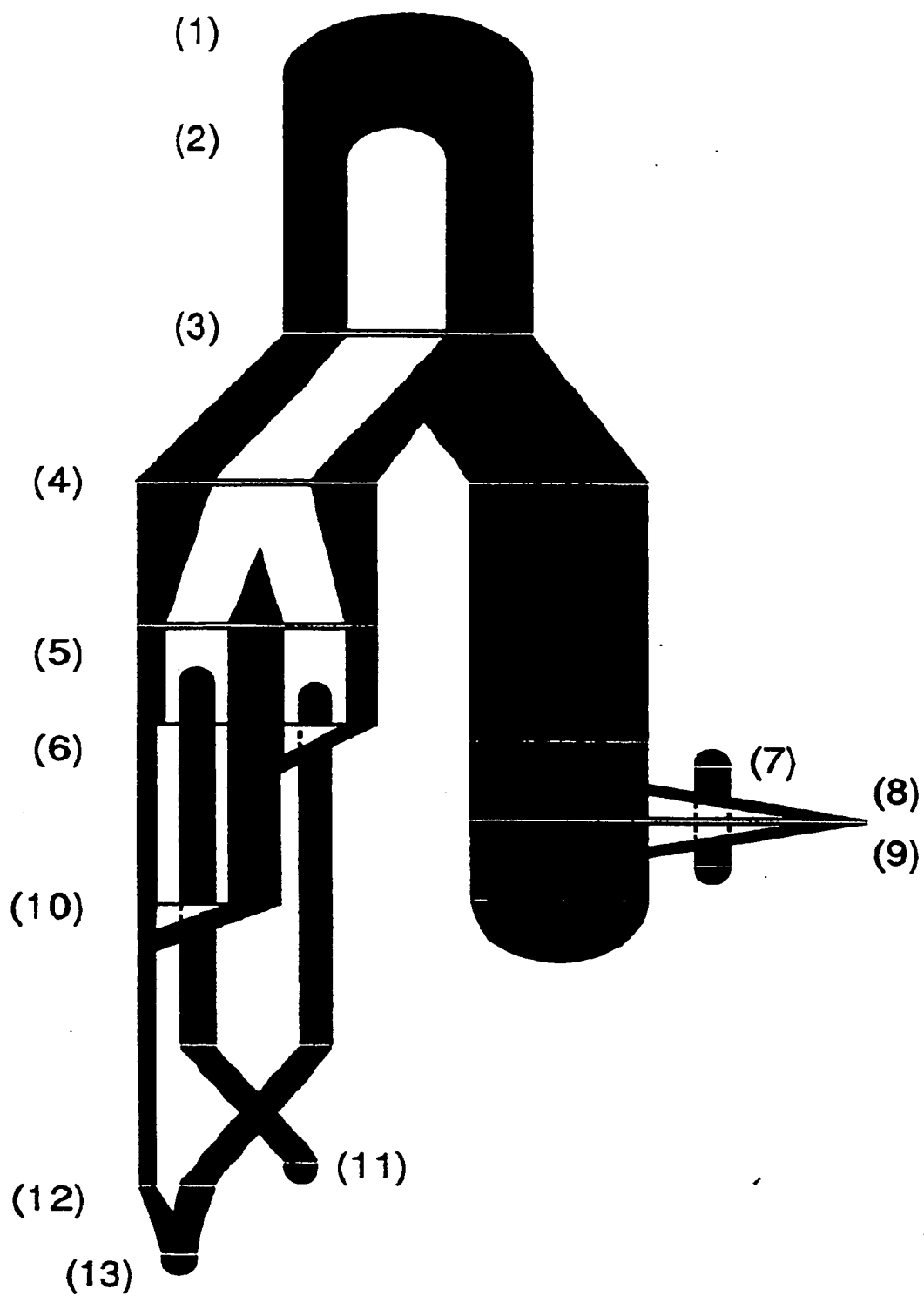


Fig. 11

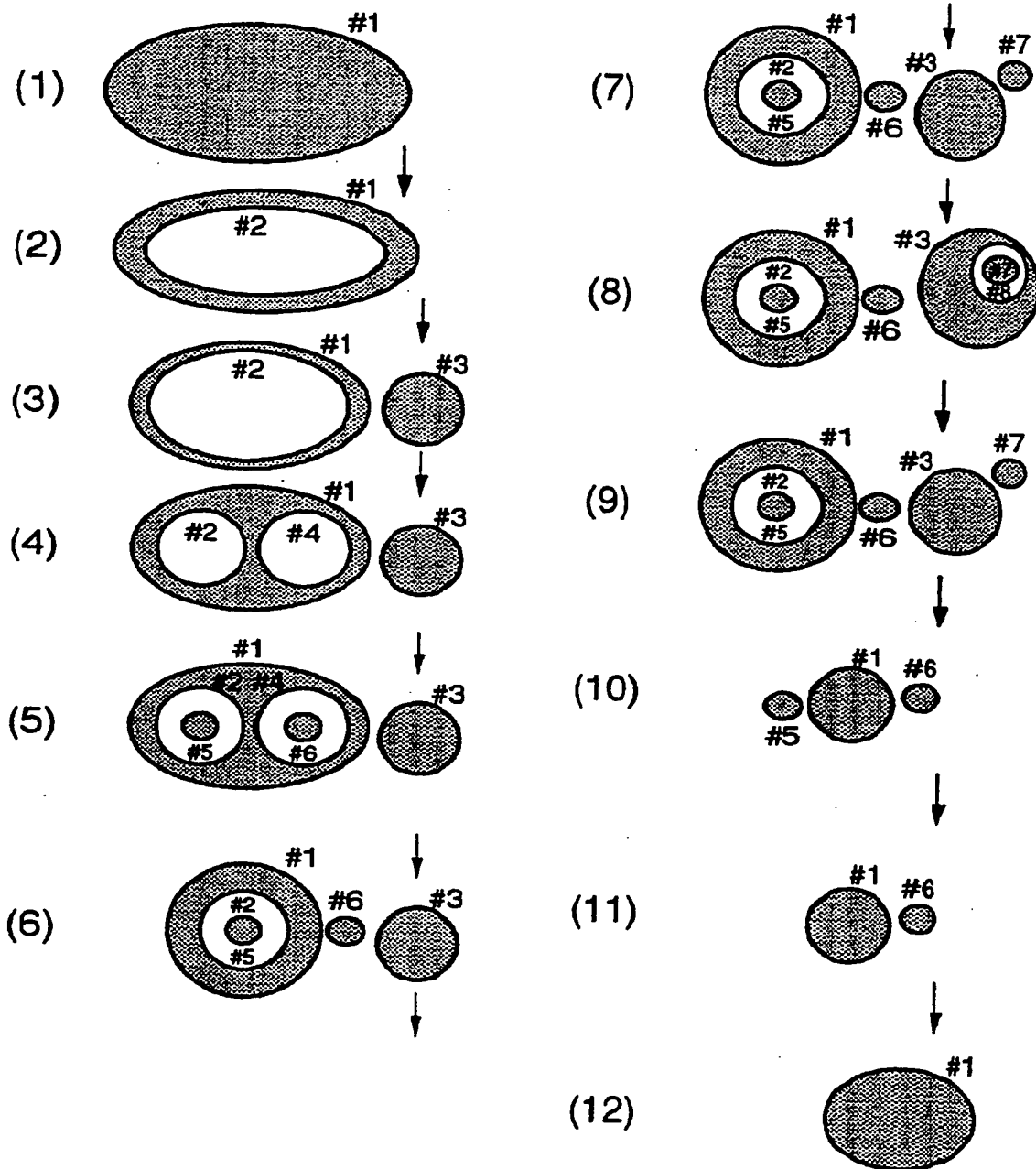


FIG. 12

1. PUT\_E2(0);
2. PUT\_E2(1);
3. PUT\_E1\_DIVIDE(1, nil, INSIDE);
4. PUT\_E1\_DIVIDE(2, nil, INSIDE);
5. PUT\_E2(2); PUT\_E2(4);
6. PUT\_E1\_MERGE(1, 4);
7. PUT\_E2(0);
8. PUT\_E1\_DIVIDE(3, list\_containing\_only(7), OUTSIDE);
9. PUT\_E1\_MERGE(3, 8); PUT\_E0(7); PUT\_E0(3);
10. PUT\_E1\_MERGE(1, 2);
11. PUT\_E0(5);
12. PUT\_E1\_MERGE(1, 6);
13. PUT\_E0(1);

**FIG. 13**

00072095-104004

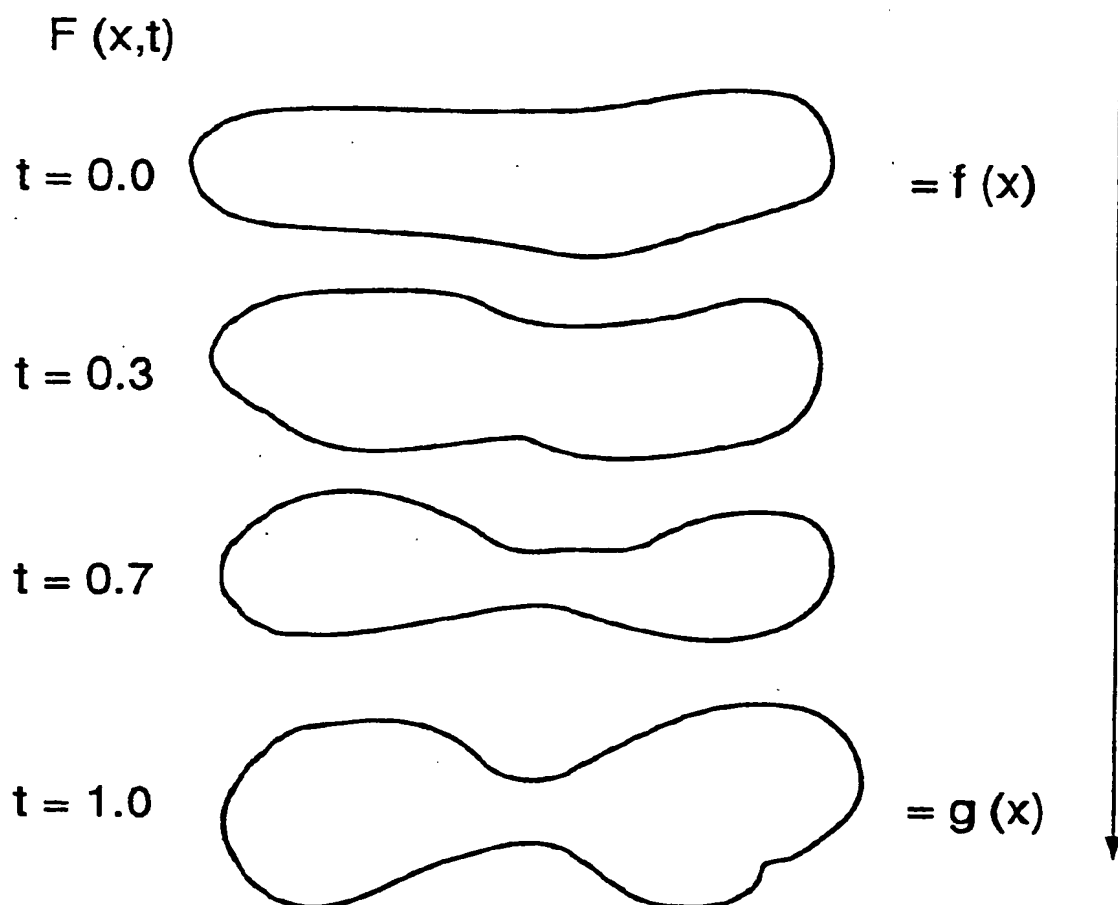


FIG. 14

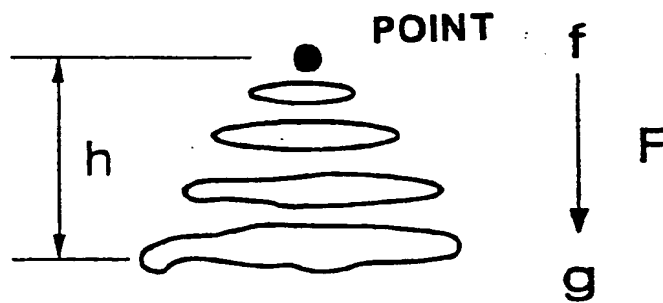


FIG. 15

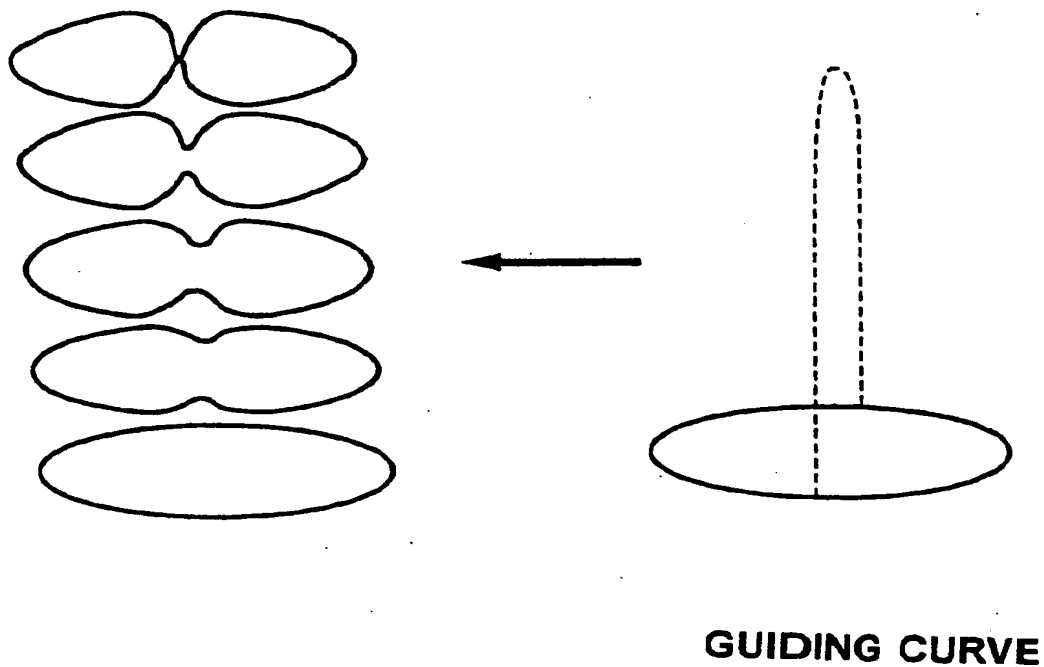
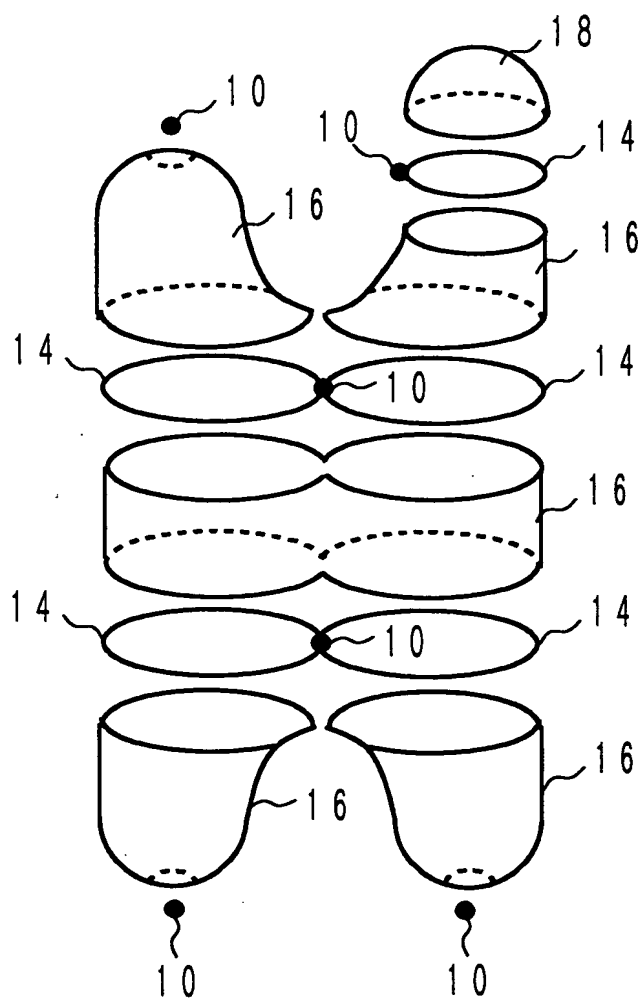


FIG. 16



**FIG. 17**

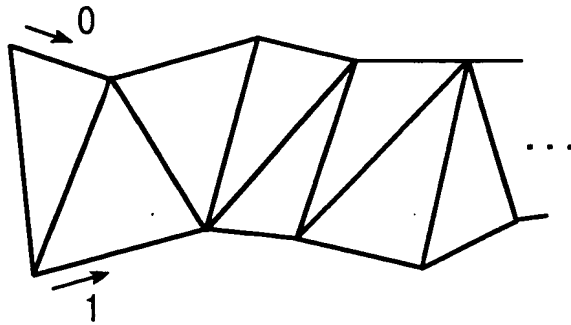


FIG.18



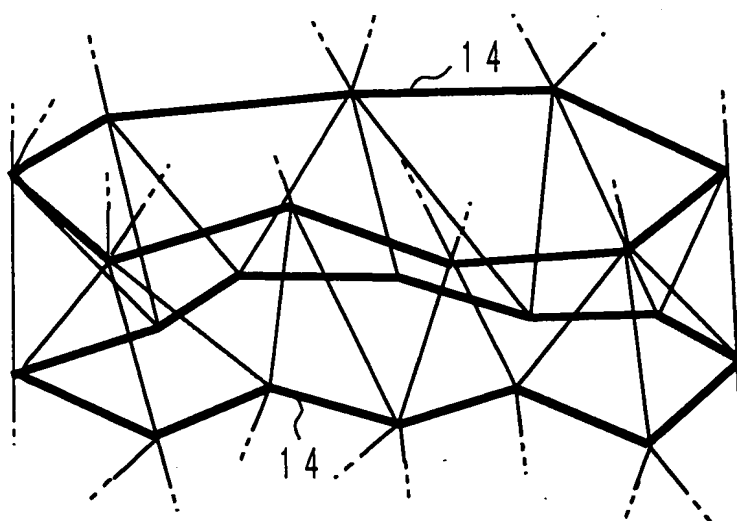
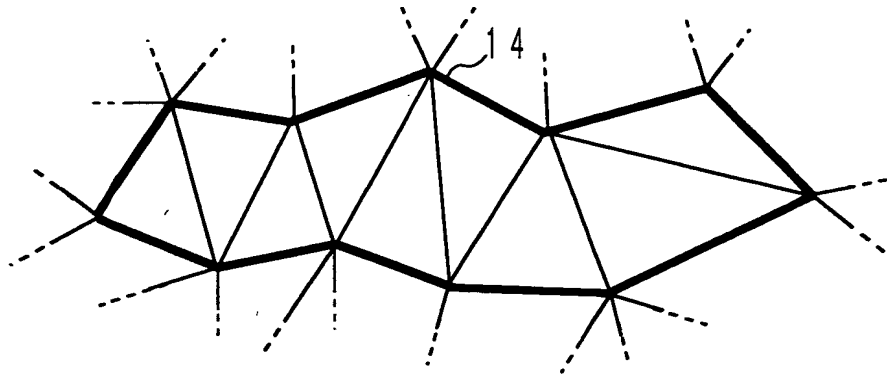
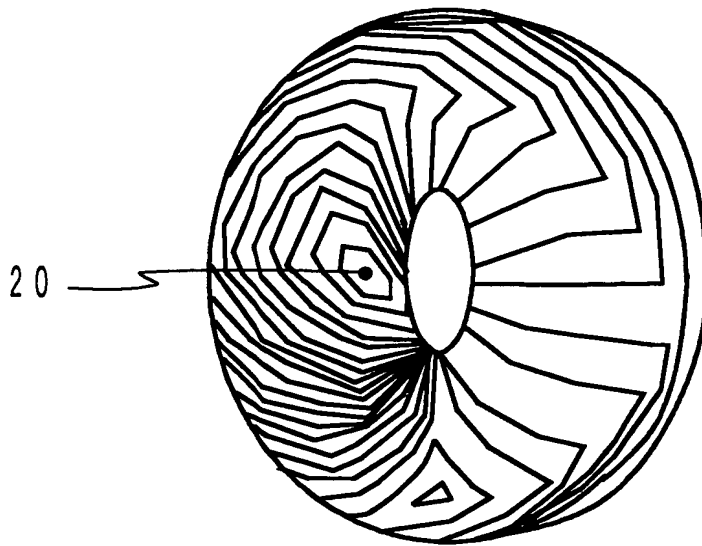


FIG.19

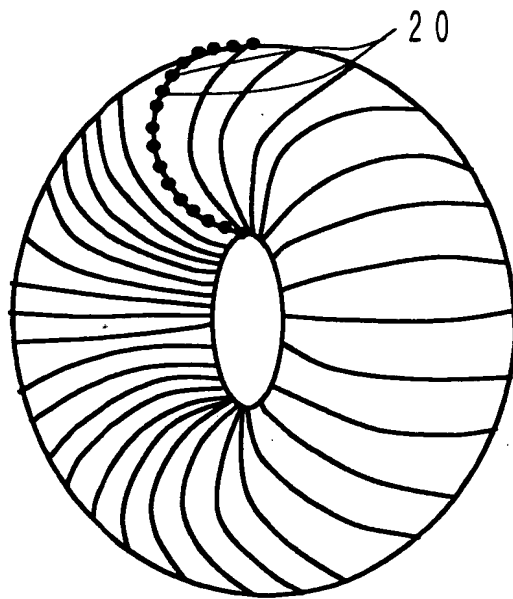


**FIG.20**



**FIG.21**

FIG. 22



**FIG.22**

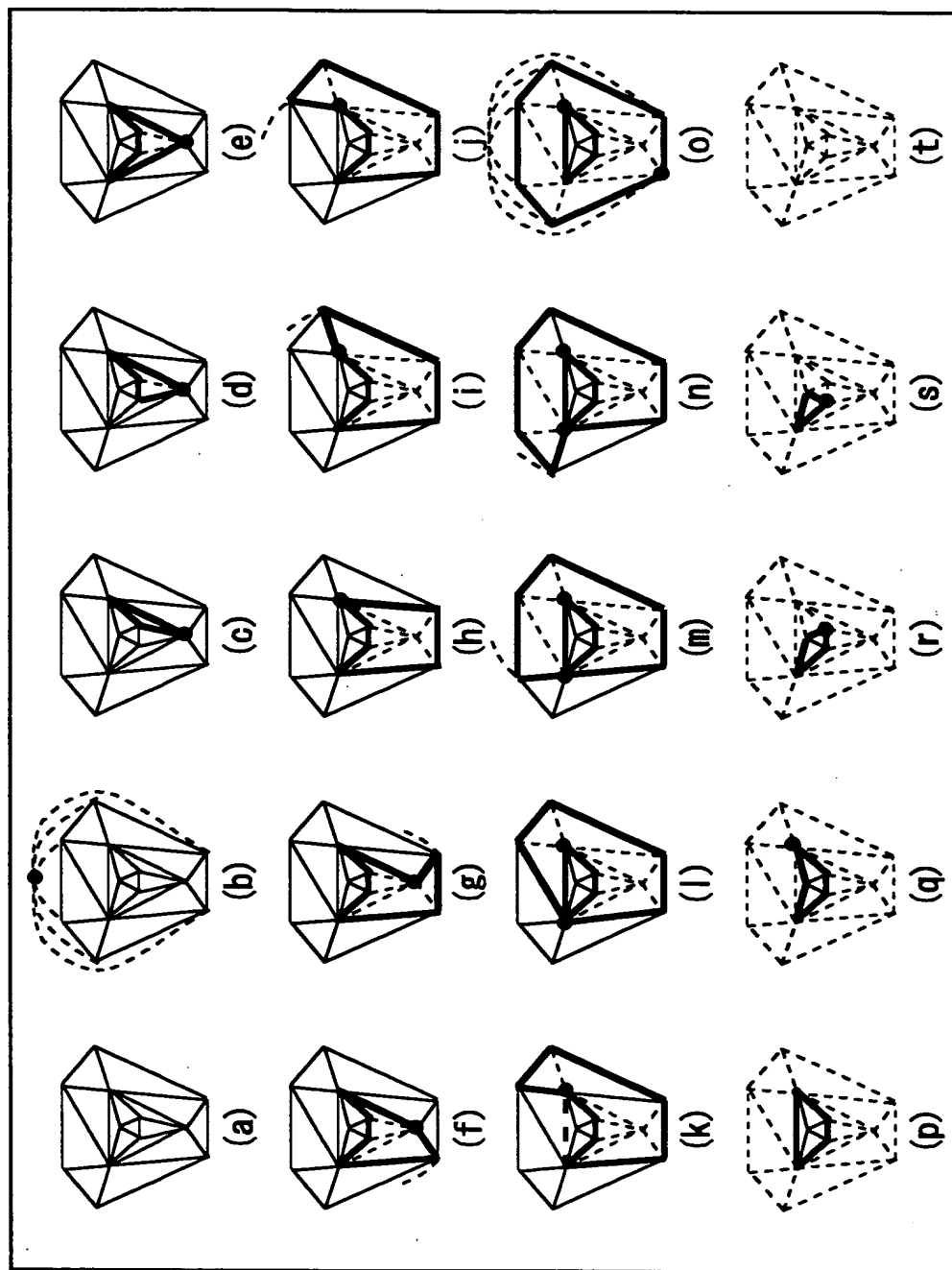
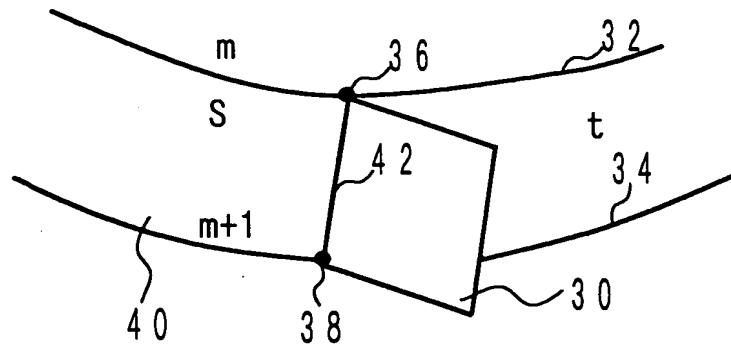


FIG.23



**FIG.24**

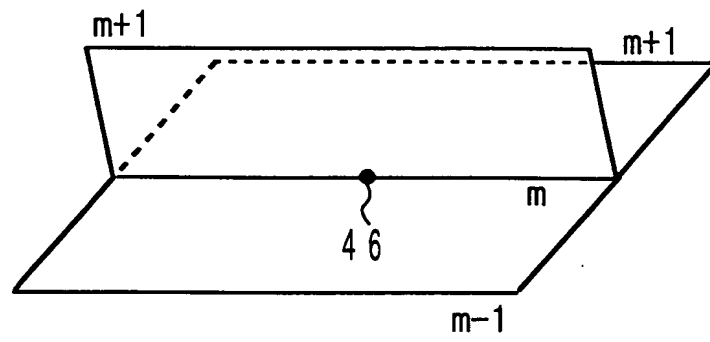


FIG.25

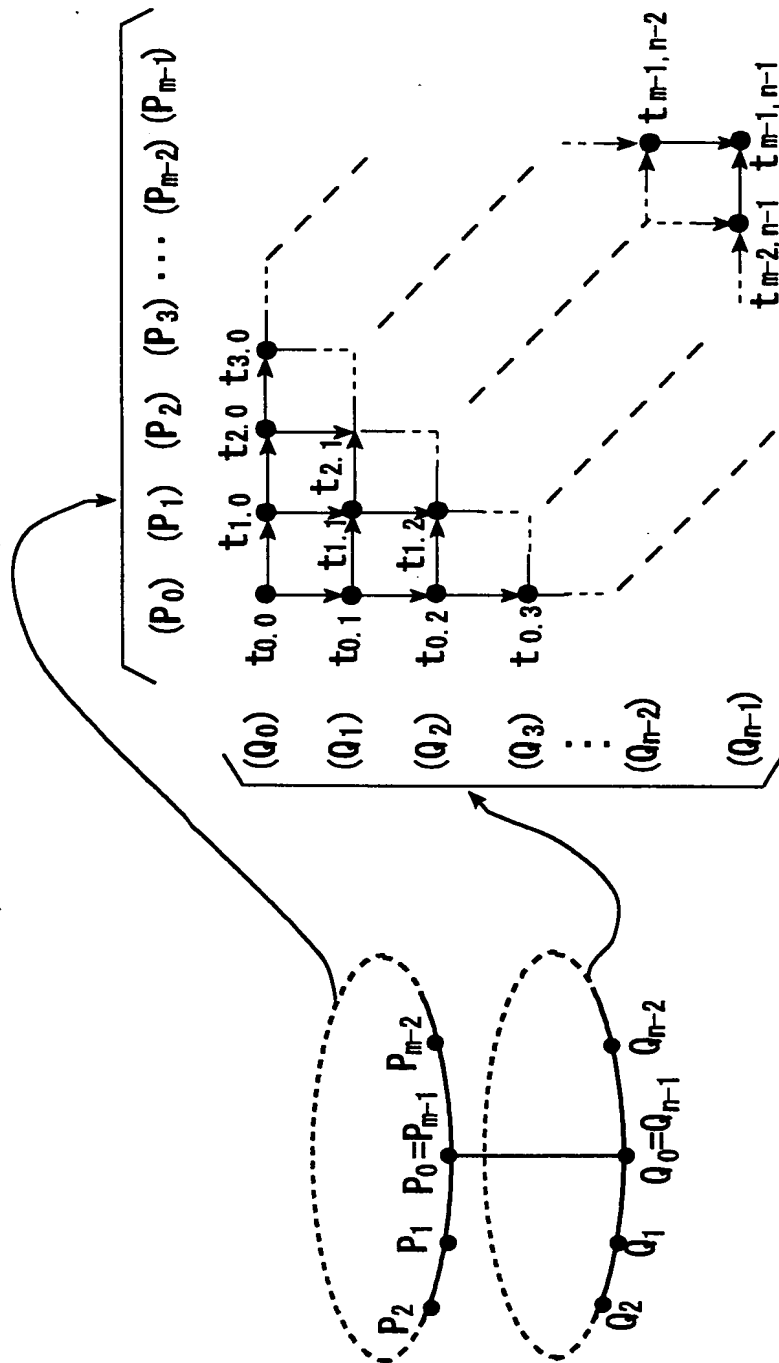
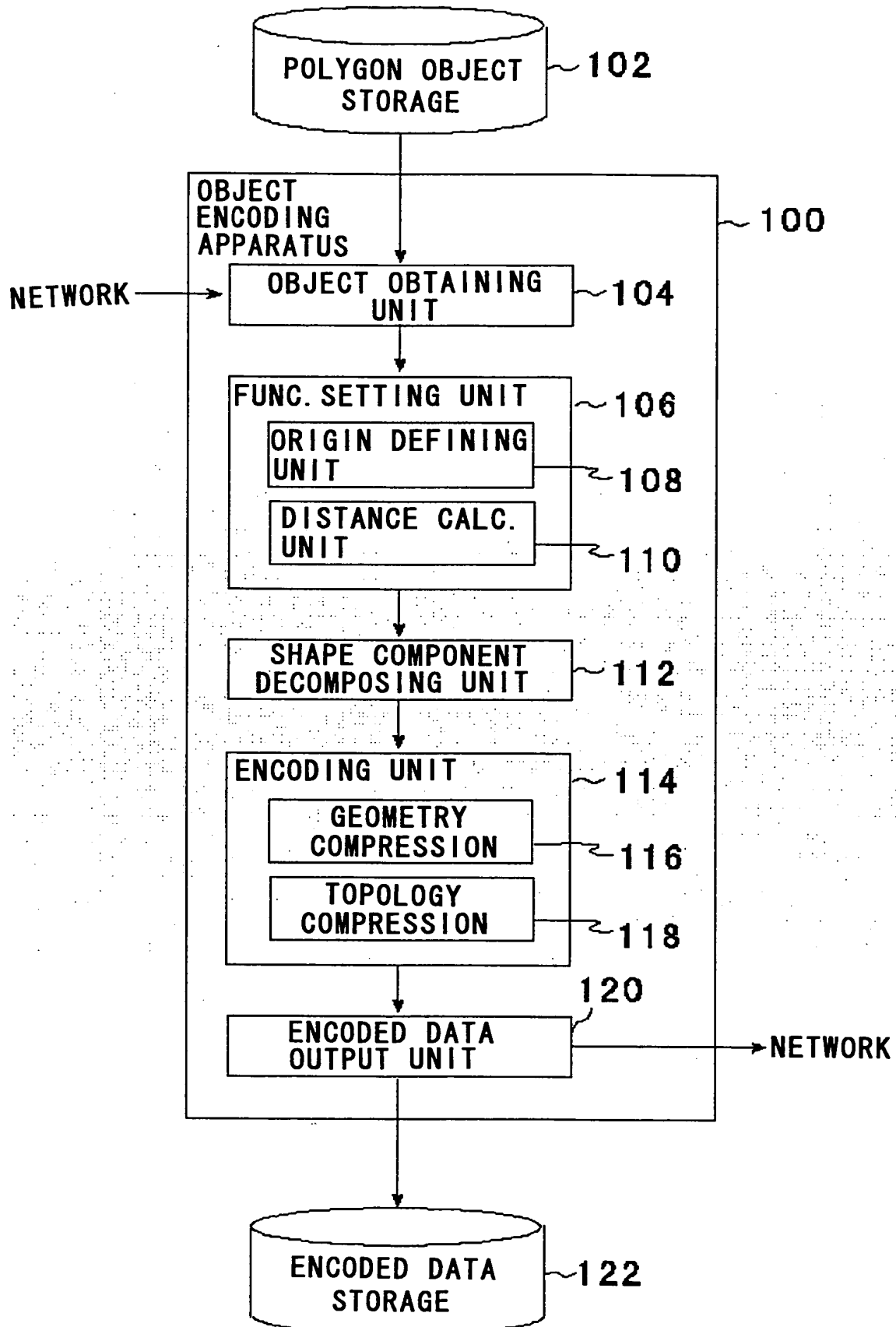


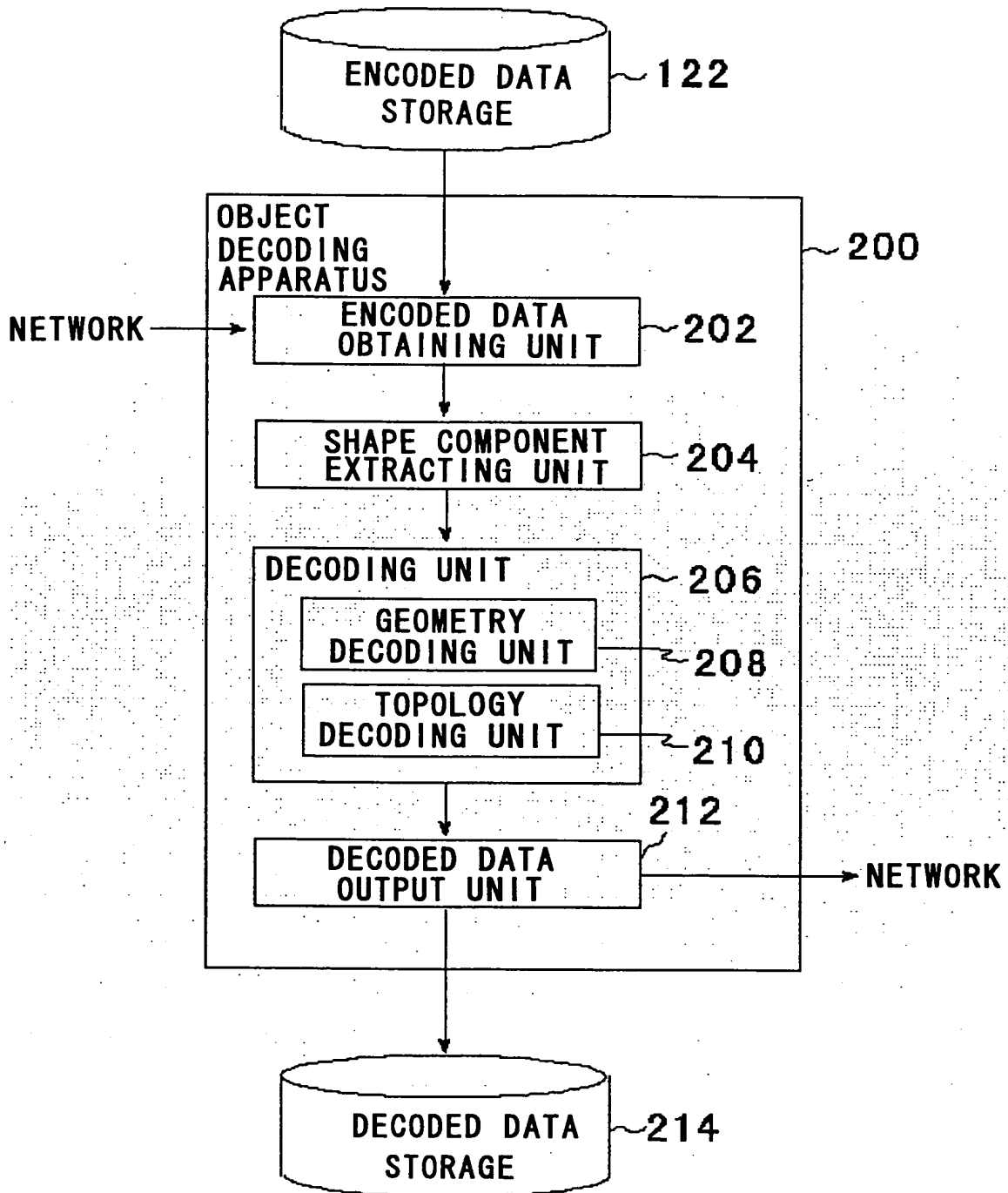
FIG.26



**FIG.27**



**FIG.28**



**FIG.29**